g) translating the second document object model document with the ApiService class as a function of the delivery technology.

22. The method of claim 21, wherein c) comprises setting a plurality of text nodes within the first document object model document to a unit of data identified by a tag in the request.

23. The method of claim 22, wherein c) further comprises limiting the unit of data to a predetermined datatype.

24. The method of claim 23, wherein c) further comprises limiting the predetermined datatype to a string.

25. The method of claim 21, wherein c) comprises setting an attribute node within the first document object model document to an attribute identified by a request name parameter in the request.

26. The method of claim 21, further comprising selecting, as a function of a mode debug flag, one of a short field name and a long field name for each of a plurality of fields in the first and second document object model documents.

27. The method of claim 21, wherein b) comprises representing an input message with the first document object model document.

28. The method of claim 21, wherein e) comprises representing an output message with the second document object model document.

29. The method of claim 21, wherein f) comprises setting, as a function of a datatype, a plurality of text nodes within the second document object model document to data read in to the second document object model document.

30. The method of claim 21, wherein f) comprises setting, as a function of a datatype, an attribute node within the second document object model document to an attribute read in to the second document object model document with the data.

31. The method of claim 30, wherein the attribute comprises an attribute name and an attribute value and f) further comprises limiting the attribute value to a predetermined datatype.

32. The method of claim 21, wherein g) comprises translating the second document object model document to extensible markup language text.

33. The method of claim 21, wherein g) comprises translating the second document object model document to at least one of a hypertext markup language and a website meta language as a function of at least one extensible stylesheet language stylesheet.

34. An e-commerce software architecture for providing a framework to interface delivery technologies with data, the e-commerce software architecture comprising:

a server computer operable to execute instructions to convert a request to an input message in a predetermined extensible markup language format, the input message comprising a plurality of request parameters,

the server computer operable to execute instructions to retrieve data as a function of the request parameters,

the server computer operable to execute instructions to create an output message in a predetermined extensible markup language format, the output message comprising the data retrieved, and

the server computer operable to execute instructions to convert the output message to a format indicated by the request.

35. The e-commerce software architecture of claim 34, wherein the request comprises a servlet request format.

36. The e-commerce software architecture of claim 34, wherein the predetermined extensible markup language format of the input message and the output message is predetermined as a function of instructions comprising a createField method and a setAttribute method.

37. The e-commerce software architecture of claim 34, wherein the instructions to convert a request to an input message comprises a createInputMessage method, a createField method and a setAttribute method.

38. The e-commerce software architecture of claim 34, wherein the instructions to create an output message comprises a createOutputMessage method, a createField method and a setAttribute method.

39. The e-commerce software architecture of claim 34, wherein the instructions to retrieve data comprises custom application code.

40. The e-commerce software architecture of claim 34, wherein the instructions to convert the output message comprises one of a generateXML method and a generatePresentation method.

41. A business services layer for leveraging extensible markup language technology to provide an interface between a back-end systems layer and a front-end systems layer, the business services layer comprising:

a server computer;

an ApiService class operable within the server computer to direct the translation of a request to an input message;

a document object model class operable within the server computer to represent the input message as a document object model document;

a Message class and a Field class operable within the server computer as wrapper of the document object model class to restrict manipulation of the document object model document; and